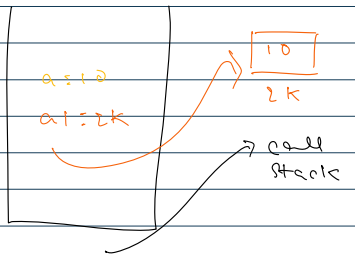
Wrapper classes

int Integer
boolean Boolean
float Float
char Character

int a = 10;
Integer a1 = 10;



boolean (primitive) ⇄ Boolean (non-primitive)
Auto-boxing →
← Unboxing

Ranges

Byte -128 to 127
Short
Integer
Long
Character 0 to 127
Boolean true or false
Float, Double ✗

Array List

resizable array

```
public class Main {
    public static void main(String[] args) {

        int a = 10;
        Integer a1 = 10; //auto-boxing

        System.out.println(a);
        System.out.println(a1);

        long l = 891;
        Long ll = 78888889l;

        a1 = a; //auto-boxing

        l = ll; //unboxing


        Integer c1 = 102;
        Integer c2 = 102;
        Integer c3 = 789;
```

```
        Integer c4 = 789;

        System.out.println(c1==c2);
        System.out.println(c3==c4);

        System.out.println(c3.equals(c4));

        Double d1 = 101.2;
        Double d2 = 101.2;
        System.out.println(d1==d2);




    }
}
```

Array List → C R U D

Create Read Updat e Delete

```
public class Main {
    public static void main(String[] args) {

        ArrayList<Integer> ll = new ArrayList<>();

        System.out.println(ll);
        System.out.println(ll.size());

        ll.add(10);
        ll.add(3);
        ll.add(20);
        ll.add(4);

        System.out.println(ll);
        System.out.println(ll.size());

        //Insert -2 at index 2
        ll.add(2, -2);

        System.out.println(ll);

        System.out.println(ll.get(3));
        // System.out.println(ll.get(5));

        System.out.println(ll.remove(3));
        System.out.println(ll);

        ll.set(1, 400);
        System.out.println(ll);

        Collections.sort(ll);
        System.out.println(ll);

        Collections.reverse(ll);
        System.out.println(ll);

    }
}
```

for-each loop

```
for(int i=0; i<ll.size(); i++){
    System.out.print(ll.get(i)+ " ");
}
System.out.println();

for(int i: ll){
    System.out.print(i+" ");
}
System.out.println();

int[] arr = new int[5];
for(Integer x: arr){
    System.out.print(x+" ");
}
System.out.println();
```

default capacity : 10 → 10 + (10/2) + 1 → 16

new capacity = old capacity + (old capacity / 2) + 1

→ 1.5x (old capacity → x)

x + x/2 + 1 = (2x + 2)/2 = 3x/2 = 1.5x

Print all substrings length wise

a k a r s h (0 1 2 3 4 5)

0 1 a 0 2 ak 0 3 aka 0 4 akar 0 5 akars 0 6 akarsh
1 2 k 1 3 ka 1 4 kar 1 5 kars 1 6 karsh
2 3 a 2 4 ar 2 5 ars 2 6 arsh
3 4 r 3 5 rs 3 6 rsh
4 5 s 4 6 sh
5 6 h

i j

j - i : 1 2 3 4 5 6

```
for(int len=1; len<=s.length(); len++){
    for(int j=len; j<=s.length(); j++){
        int i = j-len;
        sop(s.substring(i,j));
    }
}
```

j - i = len
i = j - len

```
public class Main {
    public static void main(String[] args) {
        String s = "akarsh";
        printLineWise(s);
```

```
    }

    public static void printLineWise(String s){
        for(int len=1; len<=s.length(); len++){
            for(int j=len; j<=s.length();j++){
                int i = j - len;
                System.out.print(s.substring(i, j)+" ");
            }
            System.out.println();
        }
    }

}
```

Finding CB numbers

The Challenge

Kartik Bhaiya challenges Kanak Bhaiya with a problem:

He provides a **string of digits**, and Kanak Bhaiya must determine the maximum number of **CB numbers** that can be extracted from it while following these constraints:

1. Non-overlapping Substrings:

A CB number **cannot be a substring** or **superstring** of another chosen CB number.

Example: In **4991**, both **499** and **991** are CB numbers, but we can choose only **one** of them.

2. Valid Substring Selection:

The CB number must be a **contiguous substring** of the given string.

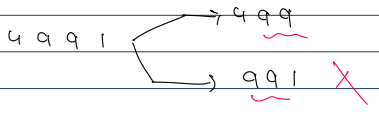
Press esc to sho

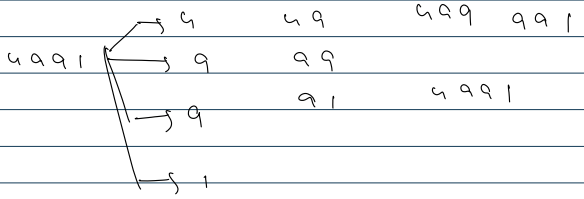
Example: In **481**, we **cannot** select **41** as a CB number because **41 is no**

3. Maximization Goal:

Since multiple solutions may exist, the goal is to find the **maximum number** of CB numbers that can be extracted from the given string.

Kanak Bhaiya has a class of Launchpad students to teach and needs help solving this challenge. Assist him in finding the solution.






```
S: "540"
int x: Integer.parseInt(S);
```

String to primitive data types.

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String s = sc.next();
        cbNumber(s);
    }

    public static void cbNumber(String s){
        int count = 0;

        boolean[] visited = new boolean[s.length()];

        for(int len=1; len<=s.length(); len++){
            for(int j=len; j<=s.length(); j++){
                int i = j - len;
                long num = Long.parseLong(s.substring(i, j));
                if(isCBNumber(num) && isVisited(visited, i, j)){
                    for(int k=i; k<j; k++){
                        visited[k] = true;
                    }
```

```
                // System.out.println(num);
                count++;
            }
        }
    }
    System.out.println(count);
}

public static boolean isVisited(boolean[] visited, int i, int j){
    for(int k=i; k<j; k++){
        if(visited[k]){
            return false;
        }
    }
    return true;
}

public static boolean isCBNumber(long n){
    if(n==0 || n==1){
        return false;
    }

    int[] arr = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
    for(int arrVal: arr){
        if(arrVal == n){
            return true;
        }
    }

    for(int arrVal: arr){
        if(n% arrVal == 0){
            return false;
        }
    }
    return true;
}



}
```